Trace Server

Igor Andjelkovic (University of Belgrade)

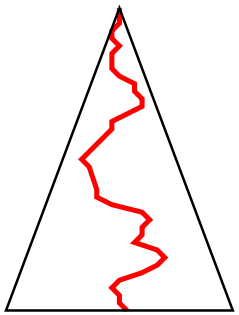
Cyrille Artho
Nat. Inst. of Advanced Industrial Science and Technology (AIST),
Tsukuba, Japan

Peter Mehlitz (NASA Ames)

Darko Marinov (University of Illinois at Urbana-Champaign)

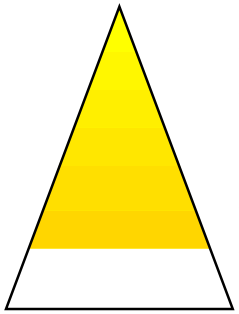
Dynamic Analysis of Software

Run-time Verification



- Analyzes few traces.
- Complex rules.
- May miss defects.
- Scalable.

Software Model Checking



- Many (all) traces.
- Simple rules.
- Finds all defects.
- Resource hungry.

Automated techniques to find defects in implementation.

Outlook

1. Out of memory — what now?
2. Solution: Trace server.
3. Convergence of technologies.
4. Google Summer of Code.
5. Demo.

Java PathFinder (JPF)

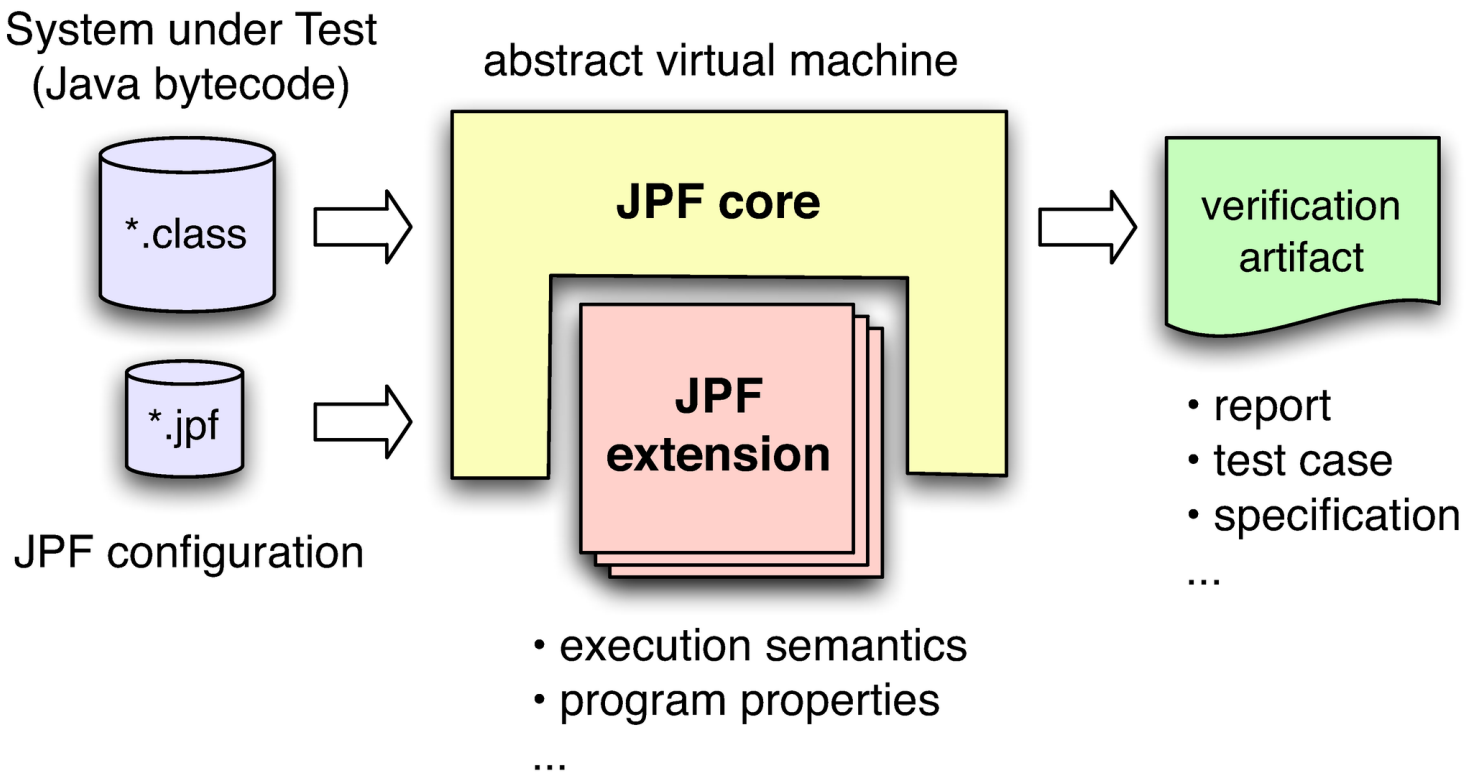


Image source: Java PathFinder tutorial,

http://babelfish.arc.nasa.gov/trac/jpf/attachment/wiki/presentations/start/jpf-graphics.zip

- Software model checker for Java bytecode.
- Extensible, now with about 20 associated projects:
  http://babelfish.arc.nasa.gov/trac/jpf/

Out of Memory...

```
[SEVERE] JPF out of memory
================================== results
no errors detected
================================== statistics
elapsed time: 5:57:54
```

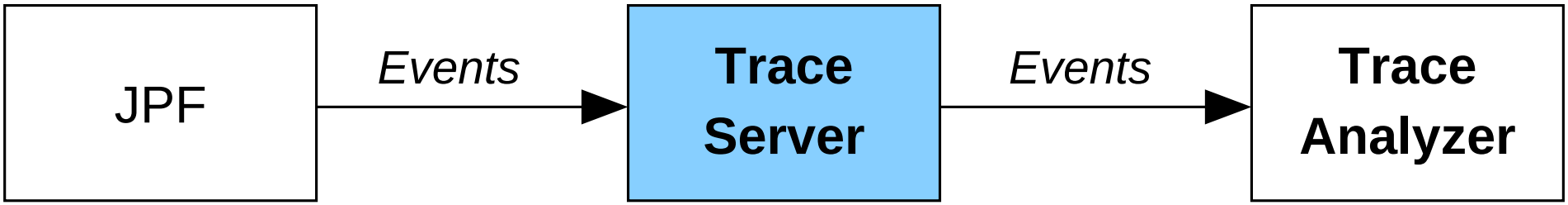
- JPF stores state space in memory for model checking.
- More memory is needed to store the trace data.
- Sometimes the sum of both is too much!

Mitigation

1. Run JPF by itself, storing only choices (no detailed trace).
2. Get result.
3. Run JPF again with stored choices just to reproduce trace.

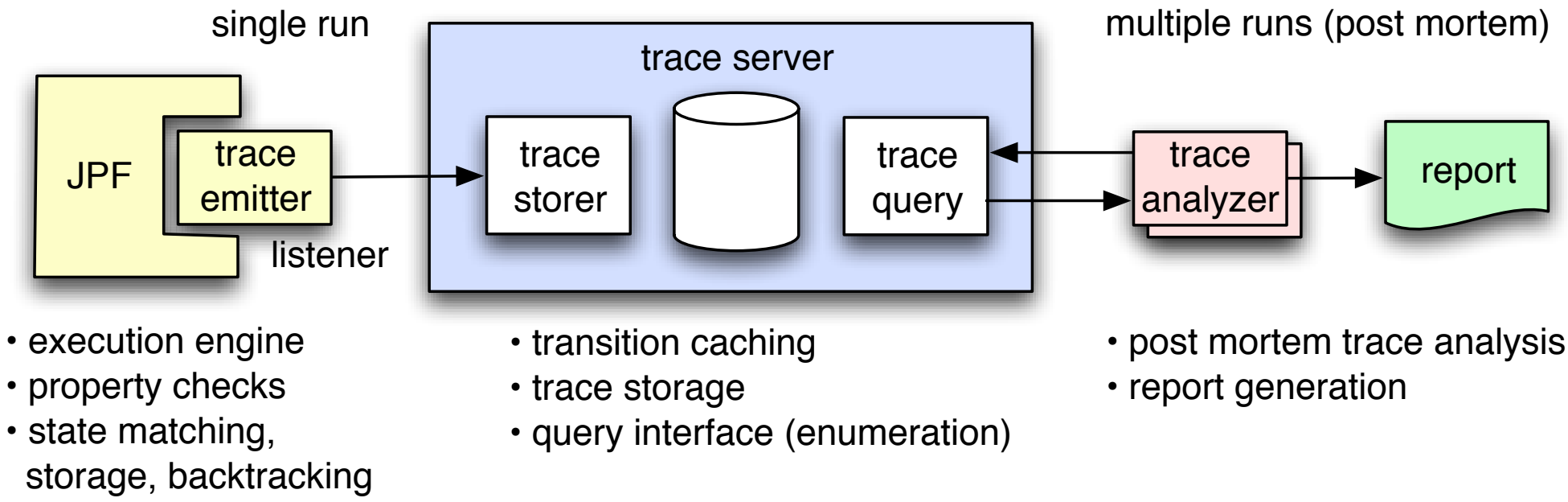
This is complicated! Can't we do better?

Better Solution



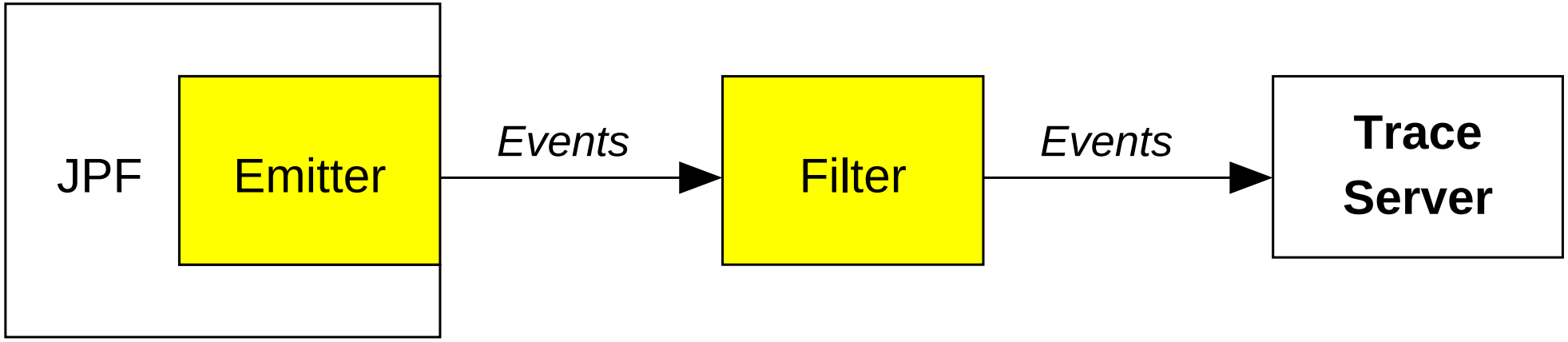
- Trace is stored outside model checker.
- Events are transmitted to **trace server** as they are generated.
- Trace server could be running on separate host.
- Analysis of trace independent of JPF.

Trace Server Architecture



- Trace emitter is designed as a JPF listener.
- Trace server hides underlying database from emitter/analyzer.
- Different analyzer algorithms may work post-mortem on trace DB.

Trace Emitter



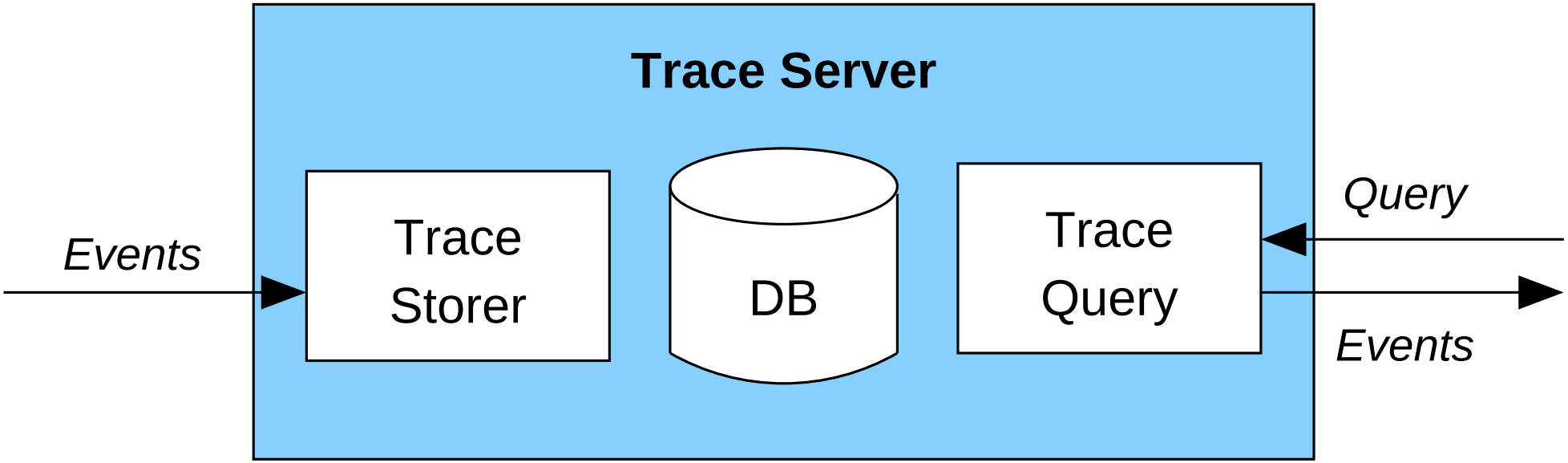
- Emitter captures events from JPF listener interface.
- Events are converted to „Property, value” pairs.
- Filter can alter/eliminate/duplicate events;
  - forwards them to another filter or
  - trace server.

Trace Filter

```
public class OnlyNewInsnFilter extends TraceFilter {

  public void processInstructionExecuted(Event ev) {

    String insnOpcode = (String)
      ev.getProperty(PropertyCollection.INSTRUCTION_OPCODE);

    if (insnOpcode.equals("new") {
      forward(ev, eventType.instructionExecuted);
} } }
```

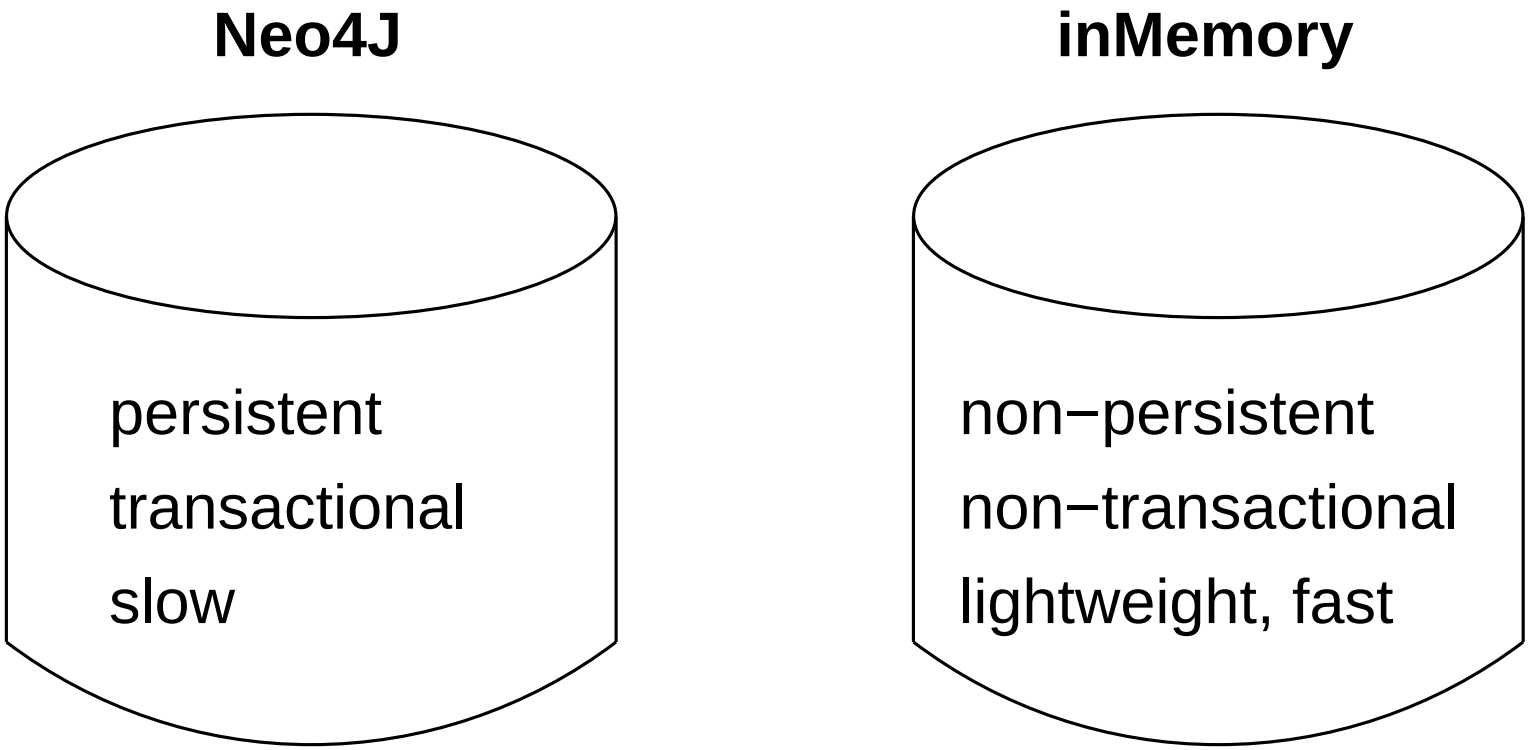
- Filter overrides default action (forward) for all instruction events.
- Only NEW instructions are retained.
- Filter may also create additional data for trace;
  for example: retain string data printed to console.

Interface to Trace Data



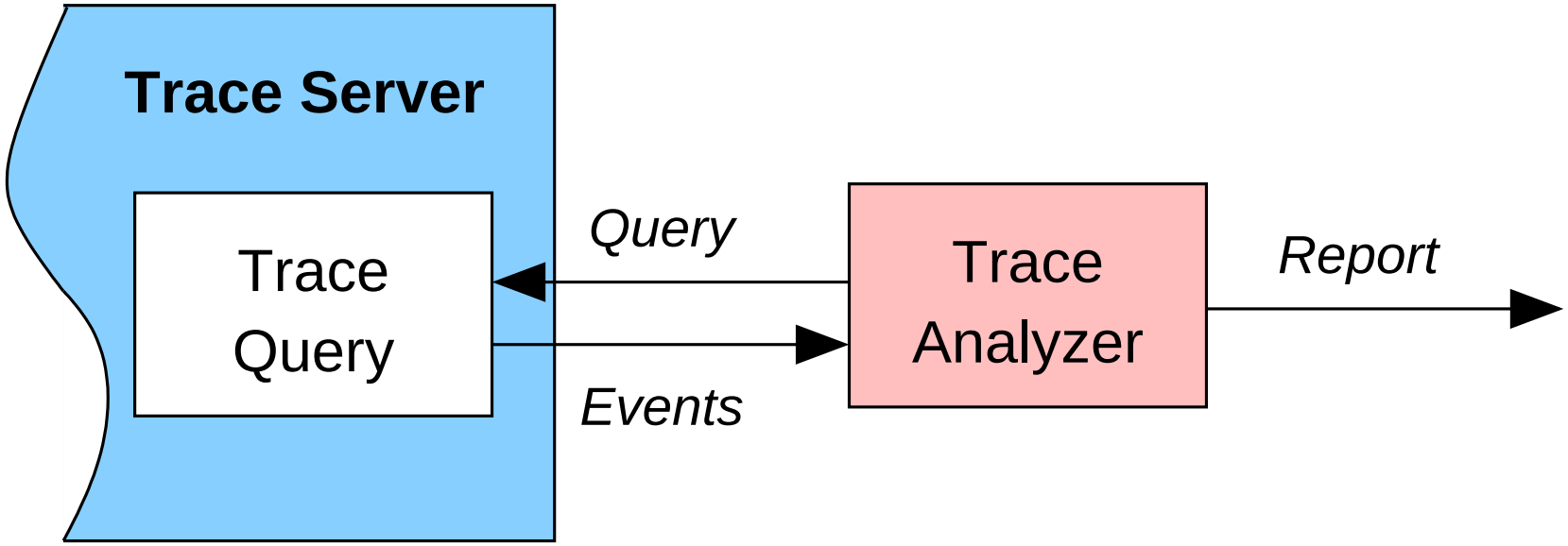
- Trace Storer provides DB-independent API to store events.
- Trace Query interface allows DB-independent querying of events:
  - Iteration over event path;
  - Filtering of events via predicates.

Trace Database



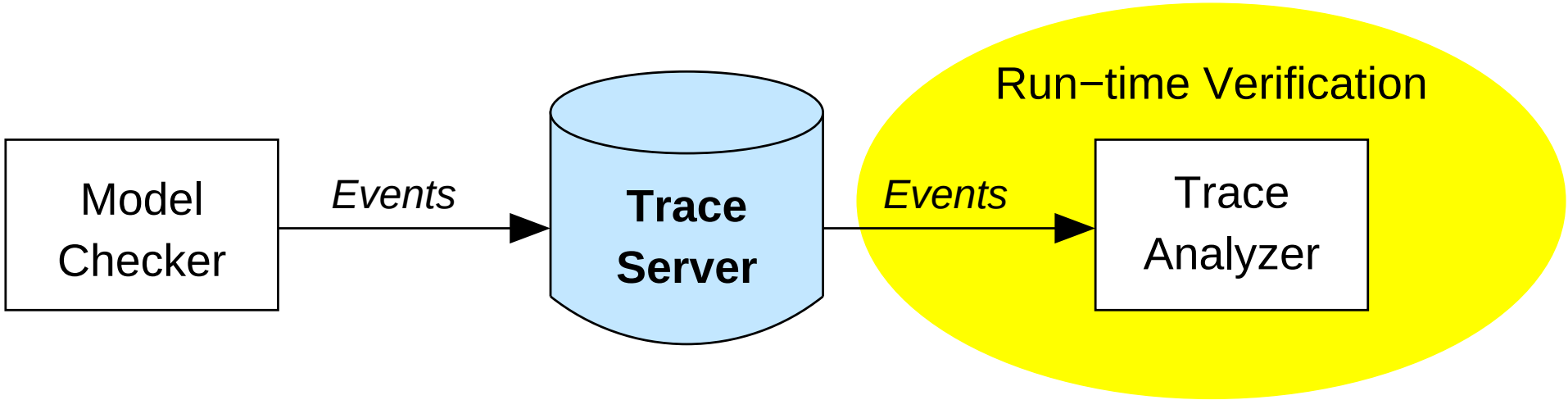
- Events stored as nodes connected by graph edges.
  - Nodes are property/value pairs for event data.
  - Values can be strings or primitive data (int, float).

Trace Analyzer



- Trace Analyzer can query data independently of DB format.
- Queries by iteration or predicate.
- Analysis of one or multiple executions.
- Data mining of trace information.

Convergence between Technologies



- Model Checker explores state space.
- Properties analyzed independently of trace generation.

Google Summer of Code

Develop open source software with sponsorship from Google.

1. Organization (NASA) applies with various candidate projects.
   - ➤ Each project has one or more mentors attached to it.
   - ➤ Mentors can be from other organizations (such as AIST).
2. Students apply to projects.
3. About 10 % of the students/30 % of the projects get chosen.
4. Students work on software over about 3 months, get USD 5,000.

Demo

To download the trace server, or for code examples, see:

http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-trace-server